SELA

AngularJS

© Copyright SELA software & Education Labs Ltd. | 14-18 Baruch Hirsch St Bnei Brak, 51202 Israel | www.selagroup.com

What Is AngularJS?

- An MV* framework for developing CRUD style web applications
- Developed by Google
- Works on all modern web browsers
- Open source (MIT license)
- No external dependencies
- Very opinionated

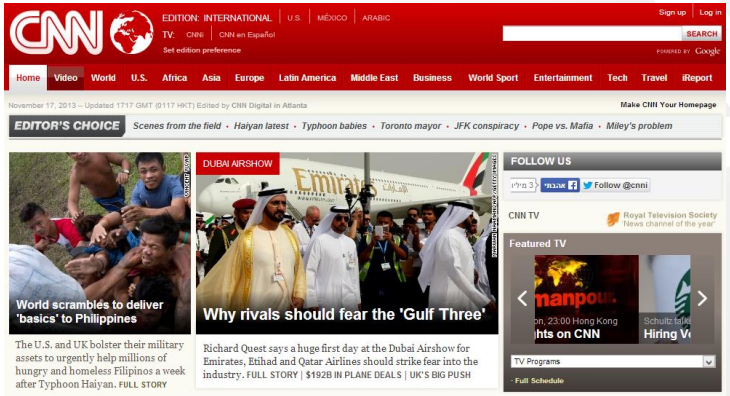
Agenda

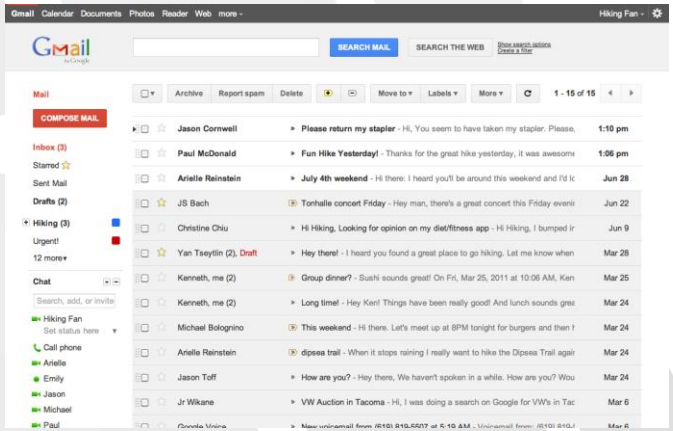
- The Modern Web
- Introduction to AngularJS
- Setting Up The Environment
- Services
- Filters
- Validation
- Routing
- Testing

The Modern Web

The Modern Web

- From web pages to web applications
- More and more logic is pushed to the client



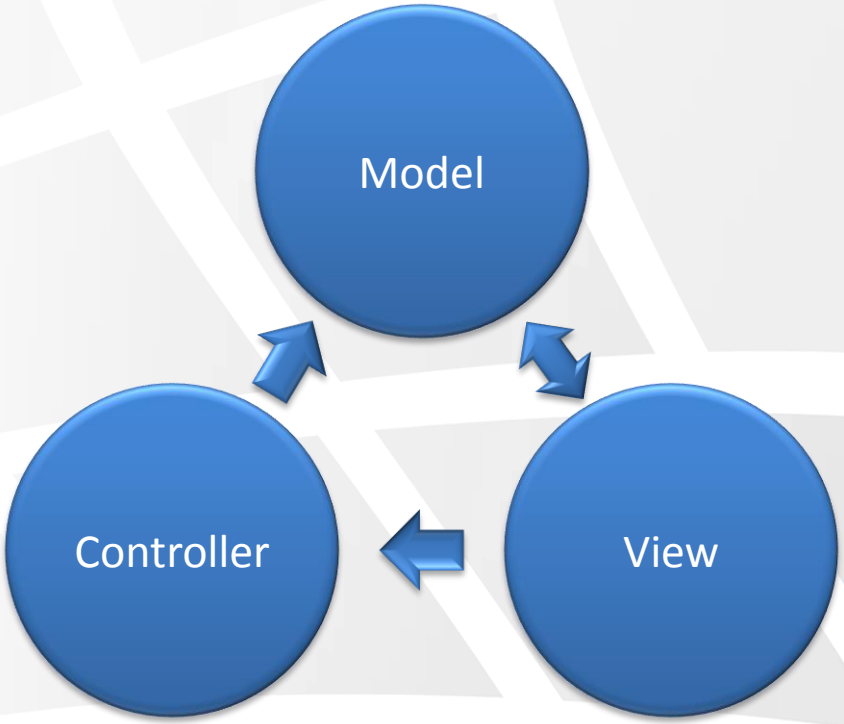


The Problem

- As we add more and more JavaScript, our application is getting:

Introduction to AngularJS

Model View Controller



MVC – The Model

- The business data
- Usually comes from a REST API
- Exposed to the view via the $scope

MVC – The View

- The HTML
- Binds to the model via the $scope
- Calls controller methods via the $scope
- Uses filters to transform the model
- Uses directives for DOM manipulation and reusability

MVC – The Controller

- Contains the business logic
- Interacts with services
- Initializes the $scope
- Exposes methods to the view
- Updates the model based on view interactions

Dissecting an Angular Application

- Below is the entire markup
- In the following slides, we'll walk through its various pieces

```
<!doctype html>
<html lang="en">
<head>
    <title>Hello Angular</title>
    <script src="bower_components/angular/angular.js"></script>
    <script src="app.js"></script>
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
          <input type="text" placeholder="Enter your name" ng-model="name">
          <button ng-click="greet()">Say Hello</button>
    </div>
</body>
</html>
```

Dissecting an Angular Application

- First, the AngularJS script should be included in the page
- The application code (app.js) is included as well and will be discussed later

```
<!doctype html>
<html lang="en">
<head>
    <title>Hello Angular</title>
    <script src="bower_components/angular/angular.js"></script>
    <script src="app.js"></script>
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
          <input type="text" placeholder="Enter your name" ng-model="name">
          <button ng-click="greet()">Say Hello</button>
    </div>
</body>
</html>
```

Dissecting an Angular Application

- The ng-app directive bootstraps the application, and instructs angular to load a module named *myModule*
- Directives are special angular components that add behavior to HTML elements.

```
<body ng-app="myModule">
    <div ng-controller="myController">
          <input type="text" placeholder="Enter your name" ng-model="name">
          <button ng-click="greet()">Say Hello</button>
    </div>
</body>
```

Dissecting an Angular Application

- The ng-controller directive associates a DOM sub tree with a specific controller
- Controllers are an angular component that provides the logic behind a specific view

```
<body ng-app="myModule">
    <div ng-controller="myController">
          <input type="text" placeholder="Enter your name" ng-model="name">
          <button ng-click="greet()">Say Hello</button>
    </div>
</body>
```

Dissecting an Angular Application

- The ng-model directive provides a two-way data-binding between a DOM element and a scope property
- The ng-click directive invokes a scope function when a DOM element is clicked

```
<body ng-app="myModule">
    <div ng-controller="myController">
          <input type="text" placeholder="Enter your name" ng-model="name">
          <button ng-click="greet()">Say Hello</button>
    </div>
</body>
```

Dissecting an Angular Application

- Below is the entire JavaScript code (app.js)
- In the following slides, we'll walk through its various pieces

```
var myModule = angular.module('myModule', []);

myModule.controller('myController', function($scope, $window){
        $scope.greet = function(){
                $window.alert("Hello " + $scope.name);
        };
});
```

Dissecting an Angular Application

- The angular.module method registers a new module named *myModule*

```
angular.module('myModule', []);

angular.module('myModule').controller('myController',
   function($scope, $window){
        $scope.greet = function(){
                $window.alert("Hello " + $scope.name);
        };
});
```

Dissecting an Angular Application

- The module.controller method creates a new controller inside of a module
- Dependencies are injected automatically

```
angular.module('myModule', []);

angular.module('myModule').controller('myController',
   function($scope, $window){
        $scope.greet = function(){
                $window.alert("Hello " + $scope.name);
        };
});
```

Dissecting an Angular Application

- A $scope is the context against which expressions in the markup are being evaluated

```
angular.module('myModule', []);

angular.module('myModule').controller('myController',
   function($scope, $window){
        $scope.greet = function(){
                $window.alert("Hello " + $scope.name);
        };
});
```

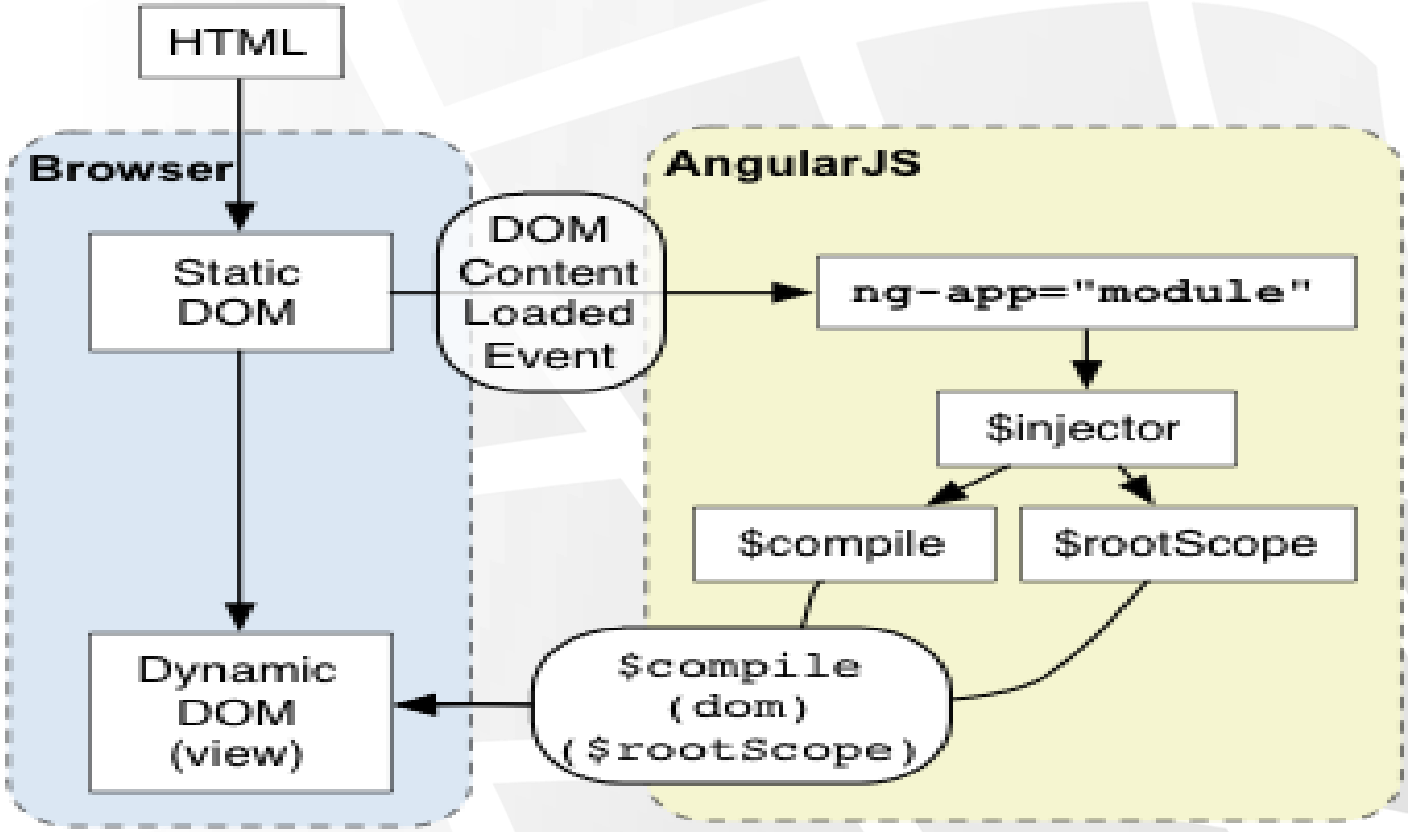
Angular Application Flow

- The browser makes an HTTP request to the server to load the initial template and scripts
- Angular loads and waits for the page to be fully loaded. Then, it searches for the ng-app directive and loads the associated module and its dependencies
- Angular compiles the template, adds listeners on DOM elements and evaluates bindings
- From now on – Angular is in charge

Bootstrapping

- There are 3 important things that happen during the angular application bootstrapping:
  - Angular creates the injector that will be used for dependency injection
  - The injector creates the root scope that will become the context for the model of our application
  - Angular compiles the DOM, starting at the element that contains the ngApp directive, processing any directives and bindings found along the way

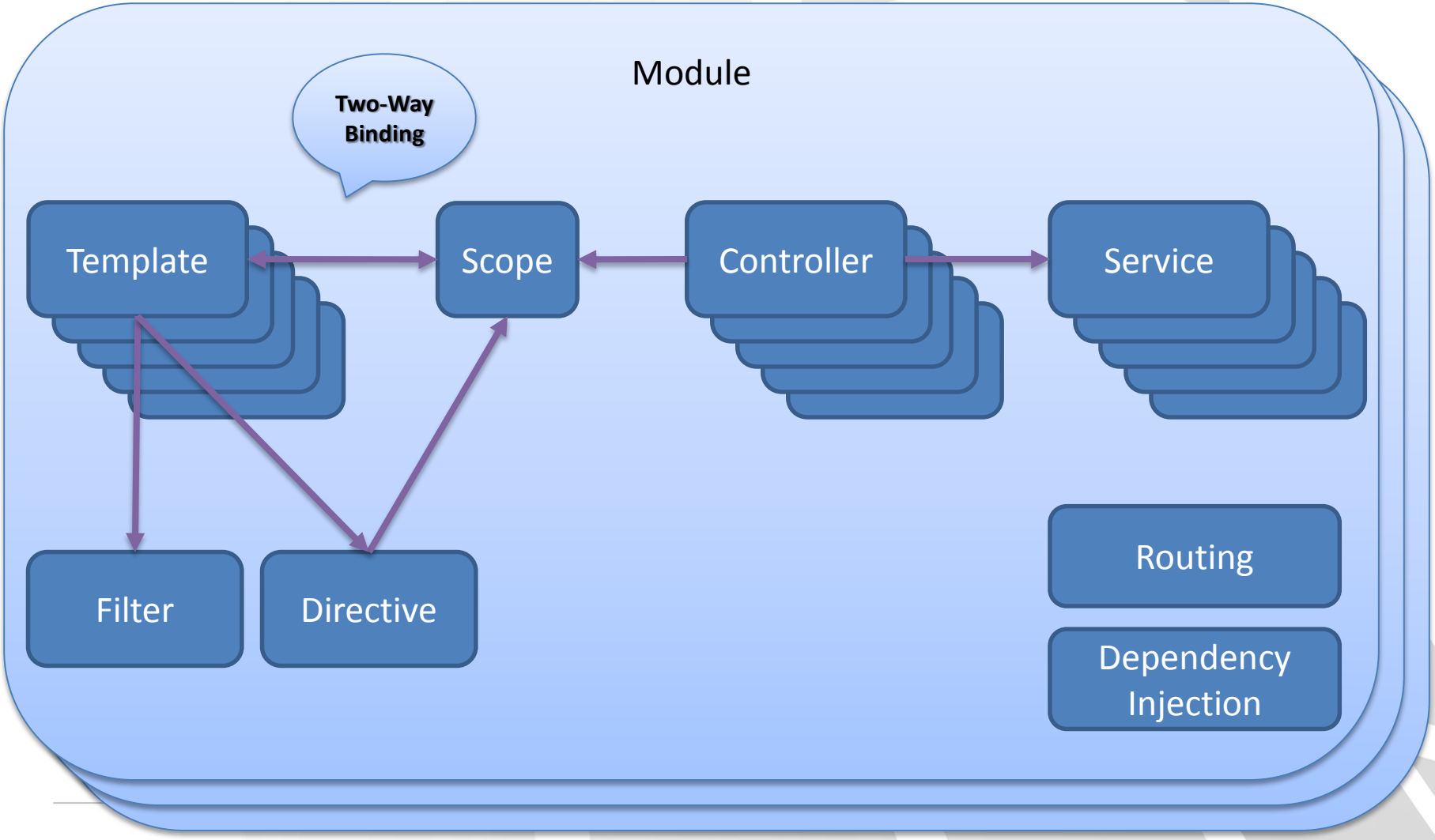
Bootstrapping Steps



Kids Bank

Demo

Angular Building Blocks



Setting Up The Environment

Getting AngularJS

- AngularJS can be downloaded from angularjs.org
- It can also be included directly from the google CDN

```
https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js
```

Getting AngularJS

- Alternatively, it can be installed via the bower package manager (requires npm)

```
npm install -g bower
mkdir myProject && cd $_
bower install angular
```

Services

The $http Service

- Used for communication with remote HTTP servers
- Based on the promise API provided by $q
- Is basically a function that receives a configuration object and returns a promise
- Provides shortcut methods for most common cases

The Configuration Object

- Describes the HTTP request to be made.
- Contains the following properties:
  - method
  - url
  - params
  - data
  - headers
  - transformRequest and transformResponse
  - xsrfHeaderName and xsrfCookieName
  - timeout
  - cache

Using The $http Service

```
// Invoke it as a function and provide a config object
$http({method: ‘GET‘, url: '/api/questions'})
   .success(function(data, status, headers, config) {
        // Called on success
   })
   .error(function(data, status, headers, config) {
        // Called on error
   });

// Or use one of the six shortcut functions
$http.get('/api/questions').success(…).error(…);
$http.post('/api/questions', question).success(…).error(…);
$http.put(('/api/questions', question).success(…).error(…);
$http.delete('/api/questions/1').success(…).error(…);
$http.head('/api/questions').success(…).error(…);
$http.jsonp(‘question?callback=callback').success(…).error(…);
```

Kids Bank

Demo

Filters

The Purpose of Filters

- Formats the value of an expression for display to the user
  - {{ expression | filter }}
  - {{ 10 | currency }} will display 10$
- May have arguments
  - {{ price | number:2}}
- Filters are chainable

Single-Value Filters

- currency – Formats a number as a currency
- date – Formats date to a string
- number – Formats a number as string
- uppercase – transforms a string to uppercase
- lowercase – transforms a string to lowercase

Using Single-Value Filters

```
// JavaScript
$scope.amount = 150;
$scope.dueDate = Date.now();
$scope.firstName = "John";
$scope.lastName = "Doe";
$scope.average = 99.12432;

// HTML
<p>Currency: {{amount | currency}}</p>
<p>Date: {{dueDate | date:'yyyy-MM-dd HH:mm:ss'}}</p>
<p>Lowercase: {{firstName | lowercase}}</p>
<p>Uppercase: {{lastName | uppercase}}</p>
<p>number: {{average | number:2}}</p>
```

Currency: $150.00

Date: 2014-02-16 18:01:53

Lowercase: john

Uppercase: DOE

number: 99.12

Array Filters

- filter – Returns a subset of the array with only the items that matches a given predicate
  - The default predicate performs a substring match. It can be overridden
  - The string can be matched against all the properties of the element or just specific properties
- orderBy – Orders the array by a element's property
- limitTo – Displays only N items from the array
  - Negative numbers are used to indicate that the last N elements should be displayed

Kids Bank

Demo

Validation

Built-In Validation Directives

- Angular provides the following validation directives:
  - required – Checks that a value exists
  - min – Checks that a value is greater than a given value
  - max – Checks that a value is lower than the given value
  - minlength – Checks that a value is longer than a given value
  - maxlength – Checks that a value is shorter than a given value
  - Pattern – Checks that a value matches a is given regular expression
- All of the above directives sets a validation error identified by their name when the condition is not met

Using the form and ngModel Directives

- Note the usage of the novalidate attribute to disable the browser's built-in validation

```
<form name="form" novalidate>
    <div>
        <input type="text" name="title" ng-model="question.title" required/>
    </div>
    <div>
        <textarea name="content" ng-model="question.content" required>
        </textarea>
    </div>
    <button ng-click="addQuestion(question)”>
        Submit
    </button>
</form>
```

Form State

- The state of each control and of the entire form is automatically managed by Angular
- There are 4 states:
  - $pristine – No interaction has been made yet
  - $dirty – Interaction has been made
  - $invalid – There are some validation errors
  - $valid – There are no validation errors

Binding to State

- Note the usage of the ng-show and ng-disabled directives to hide or disable controls based on the form's, or its elements', state

```
<form name="form" novalidate>
    <div>
        <input type="text" name="title" ng-model="question.title" required />
        <div ng-show="form.title.$dirty && form.title.$invalid">
            ...
        </div>
    </div>
    <button ng-click="addQuestion(question)" ng-disabled="form.$invalid">
        Submit
    </button>
</form>
```

Presenting Validation Errors

- Validation errors are exposed via the $error property of the form or a specific control
- The $error property is a object-hash, containing the state of each validator (true, false)

Binding to Validation Errors

- Note the usage of the ng-show to bind to the $error property bag

```
<form name="form" novalidate>
    <div>
        <input type="text" name="title" ng-model="question.title" required />
        <div ng-show="form.title.$dirty && form.title.$invalid">
            <span ng-show="form.title.$error.required">
                Title is mandatory
            </span>
        </div>
    </div>
</form>
```

CSS Classes

- Angular automatically sets CSS classes on the form and input elements:
  - ng-valid
  - ng-invalid
  - ng-invalid-[validation name]
  - ng-pristine
  - ng-dirty
- These classes can be used to change the style of the element according to its state

Using CSS Classes

- Setting the background color of the form's input elements to #FA787E to signal an invalid state

```
input.ng-invalid.ng-dirty {
   background-color: #FA787E;
}

textarea.ng-invalid.ng-dirty {
   background-color: #FA787E;
}
```

Kids Bank

Demo

SPA – Routing

Routing In Single Page Applications

- Unlike traditional web sites, in SPAs, the responsibility for rendering the view is on the client side
- We do, however, want to give the user the same features he is used to, like:
  - Using the browser's navigation buttons
  - Using the address bar for navigation
  - Bookmark specific pages
- How can we change the address bar without causing the browser to issue a new request?

The Pound (#) Sign

- The pound sign is called the URL-hash and used by web browsers for in-page bookmarking
- Changing what comes after the pound sign is handled entirely by the browser, and does not reload the page
- In SPAs, we can leverage it to achieve the goals described in the previous slide without issuing a server request

HTML5 History API

- Starting with HTML5, the contents of the browser's history stack can be manipulated through the pushState and replaceState APIs, exposed via the window.history object
- Using these APIs allows us to change the browsers address bar without reloading the page
- The new URL can be any URL in the same origin as the current URL
- These APIs are supported in all the modern browsers

The ngRoute Module

- Angular comes with a built-in router
- The router is packaged in its own module, named ngRoute
- To use the router, perform the following steps:
  - Install and reference the angular-route script in the HTML
  - Add the ngRoute module as a dependency to your module

The $location Service

- An abstraction on top of the window.location object
- Synchronized with the browser address bar and allows to watch or manipulate the URL
- Seamless integration with the HTML5 History API. Links are automatically rewritten to reflect the supported mode

```
<a href="/page1?id=123">link</a>
```

/index.html#/page1?id=123

/page1?id=123

Using the ngRoute Module

- Referencing the angular-route module

```
<script src="angular-route.js"></script>
```

- Adding a dependency on the ngRoute module

```
var myModule = angular.module('myModule', ['ngRoute']);
```

Route Registration

- Routes are registered in the module's config function, by calling $routeProvider.when with a path and a route object
- A default route can be registered with the otherwise method
- The contents of the route object is discussed in the following slides

```
myModule.config(function($routeProvider) {
   $routeProvider.when("/page1", {…})
                 .when("/page2", {…})
                 .when("/page3", {…})
                 .otherwise({…});
});
```

The Route Configuration Object

- Each route configuration object contains the following properties:
  - template / templateUrl – An HTML string, or a path to an HTML file, to be used by the ngView directive
  - controller – controller (or a registered controller's name) to associate with the template's scope
  - redirectTo – A name of a route to redirect to

The Route Configuration Object

```
myModule.config(function($routeProvider) {
   $routeProvider
        .when("/page1", {
                templateUrl:"partials/page1.html",
                controller: 'myController‘
        }).when("/page2", {
                templateUrl: "partials/page2.html“
        }).when("/page3", {
                templateUrl: "partials/page3.html“
        }).when("/page4", {
                redirectTo: "/page3“
        }).otherwise({
                templateUrl: "partials/page1.html",
                controller: 'myController‘
        });
});
```

The ngView Directive

- The ngView directive marks the place in which the new route's template should be rendered
- Can be used as an element, or as an attribute on any element

```
<body ng-app="myModule">
     <div>
          <ul>
               <li><a href="page1">Page 1</a></li>
               <li><a href="page2">Page 2</a></li>
          </ul>
     </div>
     <div ng-view/>
</body>
```

Parameterized Routes

- A route's path can contain multiple named parameters
- A parameter is prefixed with a semicolon. For Example:

```
/users/:id/orders/:orderId
```

- The parameters can be accessed via the $routeParams service by their name
- The $routeParams service contains the query-string parameters as well

Using Parameterized Routes

- Registering a parameterized route:

```
$routeProvider.when("/page3/:id/:name", {
        templateUrl: "/partials/page3.html",
        controller: 'myController'
   });
```

- Accessing the route parameters:

```
myModule.controller('myController', function($scope, $routeParams){
   $scope.name=$routeParams.name;
   $scope.id=$routeParams.id;
});
```

Kids Bank

Demo

Summary

- Angular is an MV* framework developed by google
- It contains various components that reduces the complexity of creating web applications
- It can be downloaded directly from the AngularJS site, or installed via bower or Yeoman

Questions?